How to setup qua-kit on your local machine?

Set up local database with Postgres

- install postgres (postgres.app and pgAdmin4)
- use qua-kit configurations (username:qua-kit, pw:qua-kit) qua-kit database configuration

Create postgres database

```
# in bash
su qua-kit #change it to user qua-kit
#su - luha # change back to the admin users
psql #start postgres
# create new db named qua-kit
\c qua-kit #go to database qua-kit
\d #display infor of this db
```

Run qua-server and connect qua-kit app to database

go to path: `cd apps/hs/qua-server/`

```
# run with postgresql database
stack build qua-server --flag qua-server:postgresql

stack exec qua-server #run the local server
```

Go to local qua-kit server Now you should be able to to see the application running.

Set up exercise details in qua-kit

login as the qua-kit super user:

- username: "admin@qua-kit.hs"
- password: "make it some random thing"

Then specify the exercise details.

Download database file?

- To make the dump file and zip it: `pg_dump dbname | gzip > filename.gz #/tmp/filename.gz` (need to run it in bash not in psql)
- To unzip the dump file and use it for analysis: `gunzip –c qua–kit–dump–20161010.sql.gz(filename.gz) | psql qua–kit(dbname)`

For details please learn pg_dump

Install LUCI

Luci is a middle conversion application that is able to transform data in different application.

```
git clone https://bitbucket.org/treyerl/luci2.git
cd luci2
mvn clean install
```

How to convert obj. to .geojson with LUCI?

1.Install Blender-Luci add on Blender-Luci clone the repository to cd `~/Library/Application\ Support/Blender/2.78/scripts/addons/blenderluci` and add one more file lucipy.py

1. open blender with the .obj file Go to File > User preference > add-ons > install add-ons from file then choose the directory.
2. In Luci directory: `mvn exec:java –pl scenario` to install luci without the geometry respository
3. In blender Luci Panel. deselect auto sync, Create scenario -> upload
4. go to `http://localhost:8080` and open TCP/IP console. In the console `connect run scenario.GetList run scenario, geojson.Get ScID=?? asAttachment=true attachmentIndentation=4`
5. You can find the converted file at ~/LuciClientFiles/

Accessing PostgreSQL database in Python

After we set up the PostgreSQL database, we can move to python to perform any data analysis or wrangling. PostgreSQL can be integrated with Python using psycopg2 module. It is a popular PostgreSQL database adapter for Python.

Connecting to an existing PostgreSQL database can be achieved with:

```
import psycopg2
conn = psycopg2.connect(database="sample_db", user = "postgres",
password = "pass123", host = "127.0.0.1", port = "5432")
```

For local host, it's easier: `conn = psycopg2.connect("host='localhost' dbname='qua-kit'")`

For basic commands how to use psycopg please read:

```
# Execute a command: this creates a new table
>>> cur.execute("CREATE TABLE test (id serial PRIMARY KEY, num
integer, data varchar);")
```

```
# Pass data to fill a query placeholders and let Psycopg perform
# the correct conversion (no more SQL injections!)
>>> cur.execute("INSERT INTO test (num, data) VALUES (%s, %s)",
...      (100, "abc'def"))

# Query the database and obtain data as Python objects
>>> cur.execute("SELECT * FROM test;")
>>> cur.fetchone()
(1, 100, "abc'def")

# Make the changes to the database persistent
>>> conn.commit()

# Close communication with the database
>>> cur.close()
>>> conn.close()
```